

Программирование

Введение

Кафедра ИВТ и ПМ
ЗабГУ

2022

План

Введение в программирование

Блок-схемы

Язык программирования Python

IDE - интегрированные среды разработки

Интерактивный режим интерпретатора

Модули

Введение в язык

Литералы

Переменные

Операции

Определение типа

Ввод и вывод

Критерии качества ПО

Рекомендации и литература

Outline

Введение в программирование

- Блок-схемы

Язык программирования Python

- IDE - интегрированные среды разработки

Интерактивный режим интерпретатора

Модули

Введение в язык

- Литералы

- Переменные

- Операции

- Определение типа

Ввод и вывод

Критерии качества ПО

Рекомендации и литература

Программирование

Программирование — процесс создания компьютерных программ.

Программирование — процесс создания компьютерных программ.

Программирование = алгоритмы + структуры данных.
–Н. Вирт

Алгоритмы и структуры данных

Структура данных – программная единица, позволяющая хранить и обрабатывать множество однотипных и/или логически связанных данных.

Алгоритмы и структуры данных

Структура данных – программная единица, позволяющая хранить и обрабатывать множество однотипных и/или логически связанных данных.

Алгоритм — набор инструкций, описывающих порядок действий исполнителя для достижения некоторого результата.

таким образом программирование - это прежде всего создание алгоритмов и структур данных, а *язык программирования* - всего-лишь инструмент

Алгоритмы и структуры данных

Программирование - это прежде всего создание алгоритмов и структур данных, а *язык программирования* - всего-лишь инструмент.

Идея построения программы из трёх основных видов организации действий: *последовательности операторов*, *оператора ветвления* и *оператора цикла* актуальна с конца 1960. Языки появлялись и умирали, а эта идея до сих пор жива.

Основные способы записи алгоритмов:

- ▶ **вербальный**, когда алгоритм описывается на человеческом языке;
- ▶ **графический**, когда алгоритм описывается с помощью набора графических изображений;
- ▶ **символьный**, когда алгоритм описывается с помощью набора символов (на языке программирования, ЯП).

Вербальное представление алгоритма. Пример

В случае термического ожога глаз нужно:

1. Срочно изолировать больного от яркого света.
2. Закапать глаза 0,5% раствором дикаина, лидокаина или новокаина.
3. Провести внутреннее обезболивание (прием анальгетика).
4. Закапать глаза 30% раствором сульфацил-натрия или 2% раствором левомицетина.
5. Немедленно следовать в больницу.

Символьная запись алгоритма. Пример

```
print("Квадраты чисел")  
  
nums = [1, 2, 7, 13, 17]  
  
for i in nums:  
    print( i, " ", i**2 )
```

Outline

Введение в программирование

Блок-схемы

Язык программирования Python

IDE - интегрированные среды разработки

Интерактивный режим интерпретатора

Модули

Введение в язык

Литералы

Переменные

Операции

Определение типа

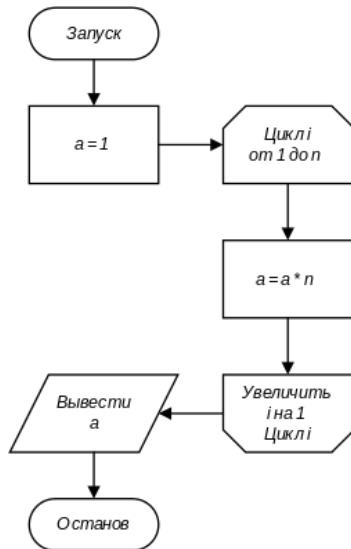
Ввод и вывод

Критерии качества ПО

Рекомендации и литература

Графическое представление алгоритма. Пример

Блок-схема



Элементы блок-схемы

ru.wikipedia.org/wiki/Блок-схема

Описание и пример

Язык программирования

Язык программирования — формальный язык, предназначенный для записи компьютерных программ.

Написанию программы в любом случае предшествует некоторая идея, постановка задачи.

Алгоритм в том или ином виде, а так же выбор структур данных - это тоже умозрительные, абстрактные понятия.

С такой точки зрения язык программирования вторичен, так как является только способом воплотить некоторую идею (алгоритм) в жизнь.

Тем более, что задача формулируется независимо от средства реализации, а алгоритм и структуры данных слабо зависят от выбора языка программирования.

Поэтому важно прежде всего уметь формулировать задачи, придумывать способы их решения и только после этого воплощать их с помощью подходящего инструмента - языка программирования.

Классификация языков программирования

Языки низкого и высокого уровня

- ▶ **Низкоуровневый язык программирования** (язык программирования низкого уровня) — язык программирования, близкий к программированию непосредственно в машинных кодах (которые исполняются процессором).
- ▶ **Высокоуровневый язык программирования** — язык программирования, разработанный для быстроты и удобства использования программистом.

Машинный код

Программа «Hello, world!» для процессора архитектуры x86
(ОС MS DOS)

```
BB 11 01 B9 0D 00 B4 0E 8A 07 43 CD 10 E2 F9  
CD 20 48 65 6C 6C 6F 2C 20 57 6F 72 6C 64 21
```

Код низкоуровневого языка

Программа «Hello, world!» на языке ассемблера `mov bx,`

```
0111h
mov cx, 000Dh
mov ah, 0Eh
mov al, [bx]
inc bx
int 10h
loop 0108
int 20h
HW db 'Hello, World!'
```

Код языка высокого уровня

Программа «Hello, world!» на языке программирования Python

```
print( "Hello, world!" )
```

Транслятор – программа преобразующая код, представленной на одном из языков программирования, в программу на другом языке.

Классификация языков программирования

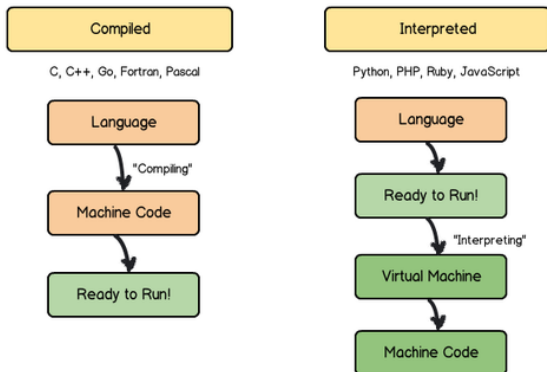
Компилируемые и интерпретируемые языки

Компиляция — трансляция программы, составленной на исходном языке высокого уровня, в эквивалентную программу на низкоуровневом языке, близком машинному коду (абсолютный код, объектный модуль, иногда на язык ассемблера)

Интерпретация — процесс одновременного чтения и выполнения исходного кода.

Классификация языков программирования

Компилируемые и интерпретируемые языки



Оранжевые этапы выполнены до запуска программы.
Зелёные этапы выполняются во время запуска программы.

Язык программирования

Язык программирования задается тремя компонентами:

- ▶ алфавитом,
- ▶ синтаксисом,
- ▶ семантикой.

Язык программирования

Алфавит – это набор различных символов: букв, цифр, специальных знаков и т. п. – то, с помощью чего записывается программа.

Например, алфавит машинного языка состоит из двух символов: 0 и 1, а если программа записана в восьмеричной системе счисления, то из восьми символов: 0, 1, 2, 3, 4, 5, 6 и 7.

Синтаксис языка программирования – набор правил, описывающий комбинации символов алфавита, считающиеся правильно структурированной программой (документом) или её фрагментом

Семантика – это смысл синтаксических конструкций.

Outline

Введение в программирование

Блок-схемы

Язык программирования Python

IDE - интегрированные среды разработки

Интерактивный режим интерпретатора

Модули

Введение в язык

Литералы

Переменные

Операции















Определение типа

Ввод и вывод

Критерии качества ПО

Рекомендации и литература

TIOBE Index

Jan 2022	Jan 2021	Change	Programming Language	Ratings	Change
1	3	▲	 Python	13.58%	+1.86%
2	1	▼	 C	12.44%	-4.94%
3	2	▼	 Java	10.66%	-1.30%
4	4		 C++	8.29%	+0.73%
5	5		 C#	5.68%	+1.73%
6	6		 Visual Basic	4.74%	+0.90%
7	7		 JavaScript	2.09%	-0.11%
8	11	▲	 Assembly language	1.85%	+0.21%
9	12	▲	 SQL	1.80%	+0.19%
10	13	▲	 Swift	1.41%	-0.02%
11	8	▼	 PHP	1.40%	-0.60%
12	9	▼	 R	1.25%	-0.65%
13	14	▲	 Go	1.04%	-0.37%
14	19	▲	 Delphi/Object Pascal	0.99%	+0.20%

Кто использует?

Google



CANONICAL

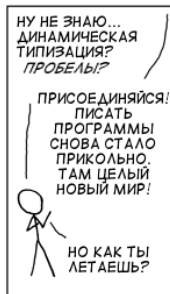
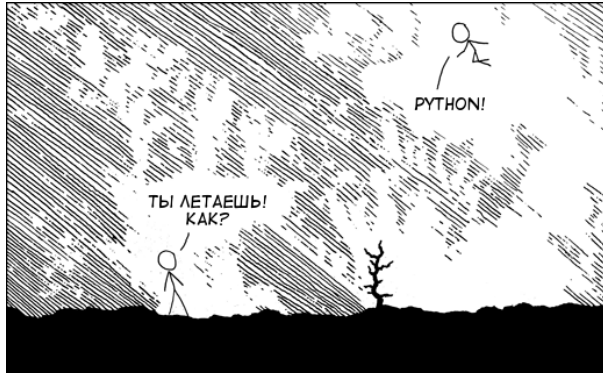
NETFLIX



YAHOO!



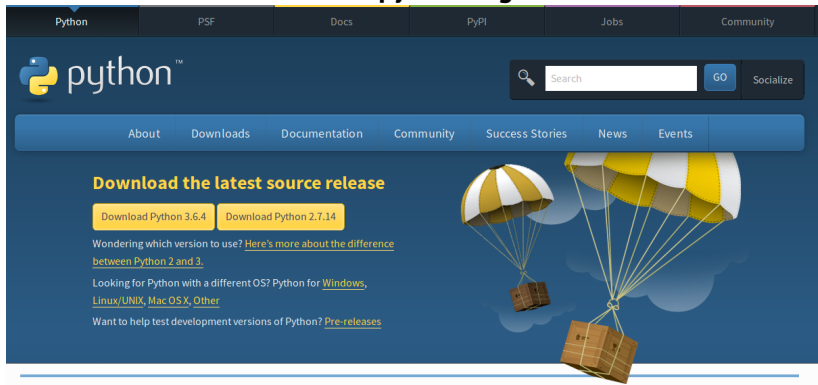
Organizations Using Python



Где скачать?

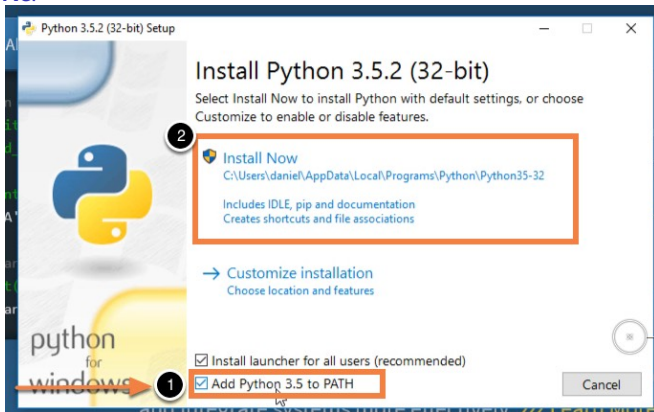
Интерпретатор и базовые средства работы с ним

www.python.org



Во многих ОС семейства Linux (например **Ubuntu**)
интерпретатор Python уже установлен.

Установка



Перед установкой лучше поставить галочку возле Add Python 3.x to PATH чтобы интерпретатор Python был непосредственно доступен из командной строки.

Стоит обратить внимание и на путь установки.

Подробнее об установке: <https://python-scripts.com/install-python-windows>

Outline

Введение в программирование

Блок-схемы

Язык программирования Python

IDE - интегрированные среды разработки

Интерактивный режим интерпретатора

Модули

Введение в язык

Литералы

Переменные

Операции

Определение типа

Ввод и вывод

Критерии качества ПО

Рекомендации и литература

Реализация языка программирования

На практике для решения большинства задач одного интерпретатора (или компилятора) недостаточно. Используется интегрированная среда разработки - IDE

Integrated Development Environment (IDE) = Транслятор +
Отладчик + редактор исходных кодов + документация + ...

- ▶ Транслятор (компилятор или интерпретатор)
- ▶ Отладчик
- ▶ Набор библиотек
- ▶ Документация

IDE - интегрированные среды разработки

- ▶ PyCharm

Удобная, функциональная, тяжеловесная.

- ▶ Wing

Простая

- ▶ Microsoft Visual Studio с плагином Python Tools

- ▶ Jupiter (Anaconda)

Хорошо подходит для небольших программ и экспериментов, похож на matlab, нет отладчика.

- ▶ Atom, SublimeText, ...

Продвинутые текстовые редакторы. Можно использовать в качестве IDE. Например: www.youtube.com/watch?v=zE6LMesKo - настройка Atom для работы с Python в ОС Windows

При выборе IDE можно ориентироваться на выбор сообщества и отрасли. см <https://stackshare.io>

PyCharm на stackshare

О Python

- ▶ Высокоуровневый язык
- ▶ Общего назначения
- ▶ Ориентирован на повышение производительности разработчика
- ▶ Интерпретируемый
- ▶ Компилируется в байт-код
- ▶ Объектно-ориентированное программирование
- ▶ Функциональное программирование
- ▶ Рефлексивность
- ▶ Динамическая типизация
- ▶ Сборка мусора

Философия

import this

- ▶ Простое лучше, чем сложное.
- ▶ Сложное лучше, чем запутанное.
- ▶ Плоское лучше, чем вложенное.
- ▶ Разреженное лучше, чем плотное.
- ▶ Читаемость имеет значение.
- ▶ Встретив двусмысленность, отбрось искушение угадать.
- ▶ Должен существовать один — и, желательно, только один — очевидный способ сделать это.
- ▶ Сейчас лучше, чем никогда.
- ▶ Если реализацию сложно объяснить — идея плоха.
- ▶ Если реализацию легко объяснить — идея, возможно, хороша.
- ▶ ...

Как работать с Python?

- ▶ В интерактивном режиме
Каждая команда (может быть составной) выполняется тут же
этот режим похож на работу в командной строке
- ▶ Программы в отдельном файле (в среде разработки)

Outline

Введение в программирование

Блок-схемы

Язык программирования Python

IDE - интегрированные среды разработки

Интерактивный режим интерпретатора

Модули

Введение в язык

Литералы

Переменные

Операции

Определение типа

Ввод и вывод

Критерии качества ПО

Рекомендации и литература

Интерактивный режим

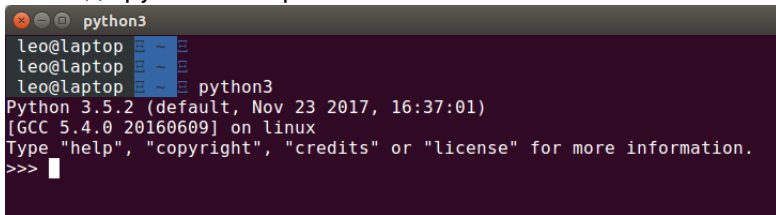
Запуск интерпретатора Python в интерактивном режиме:

- ▶ Windows

Запустить Python непосредственно или `python.exe` (из командной строки)

- ▶ Linux

Команда `python3` в терминале



```
python3
leo@laptop ~$ python3
Python 3.5.2 (default, Nov 23 2017, 16:37:01)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```


Интерактивный режим интерпретатора

Использование python как калькулятора в интерактивном режиме

- ▶ работает как командная строка
- ▶ умеет производить математические вычисления
- ▶ результат вычисления сразу отображается на экране
- ▶ результат вычисления автоматически (неявно) записывается в переменную `_` (знак подчёркивания)
- ▶ чтобы просмотреть содержимое переменной достаточно указать её имя (идентификатор)

```
>>> имяпеременной
```

Интерактивный режим

Использование python в качестве калькулятора

- ▶ Доступные арифметические операции: +, -, *, /
- ▶ Возведение в степень, x^y

```
3.14**3  
= 30.9591440000000002
```

- ▶ Целочисленное деление

```
7 // 3  
= 2
```

- ▶ Остаток от деления

```
12 % 5  
= 2
```

Outline

Введение в программирование

Блок-схемы

Язык программирования Python

IDE - интегрированные среды разработки

Интерактивный режим интерпретатора

Модули

Введение в язык

Литералы

Переменные

Операции

Определение типа

Ввод и вывод

Критерии качества ПО

Рекомендации и литература

Как сделать больше, чем простые математические вычисления?

Компилятор или интерпретатор языка программирования распространяются вместе со набором подпрограмм (функций), готовых для использования программистом.

Этот набор подпрограмм, типов данных, переменных и т.п. называется **стандартной библиотекой**.

- ▶ **Стандартная библиотека** разделена на части - **модули**.
- ▶ Каждый модуль содержит подпрограммы и другие элементы предназначенные для определённой цели. Например для математических вычислений или для работы с изображениями.
- ▶ Некоторые подпрограммы являются частью языка (а не стандартной библиотеки). Например подпрограммы вывода данных на экран **print** и ввода данных с клавиатуры **input**.

Модули

- ▶ Модули позволяют программисту воспользоваться уже готовыми подпрограммами, чтобы уже с их использованием строить свои программы.
- ▶ Чтобы воспользоваться подпрограммами модуля, для начала его нужно *подключить*.
- ▶ Для подключения модуля используется оператор (команда) **import**, следом за которой через пробел указывается имя модуля.

import module_name

Подключение модулей

чтобы использовать математические функции и константы
нужно подключить модуль `math`

```
import math
```

Использование модулей

Чтобы вызывать функцию из этого модуля следует указать его имя и уже потом, через точку, имя функции
Например:

```
import math
```

```
math.sin ( math.pi / 2 ) # 1.0  
math.degrees( pi / 2 ) # 90. радианы -> градусы.  
math.radians( 90 ) # 3.14. градусы -> радианы  
math.exp ( 2 ) # e^2 = 7.39  
math.sqrt(81) # 9
```

решётка – знак комментария.

Использование модулей

Как узнать что есть в модуле и как этим пользоваться?

- ▶ справка по модулю (в интерактивном режиме)

```
>>> import math
```

```
>>> help( math ) # покажет справку по модулю
```

```
>>> help( math.sin ) # покажет справку по функции
```

- ▶ Документация на официальном сайте
<https://docs.python.org/3/library/index.html>
- ▶ Google, stackoverflow, ...

В документации порой есть примеры. Если примеров нет, то перед использованием функцию можно опробовать в интерактивном режиме.

Использование модулей

Однако каждый раз указывать имя пакета (модуля) неудобно. Поэтому можно при его подключении указать конкретные функции, что будут использованы:

```
from math import tan, sin
```

Теперь можно использовать функции `tan` и `sin` непосредственно, без указания имени пакета:

```
sin (3.1415)
```

```
tan (3.1415 / 4)
```

Использование модулей

Если функций много, то их перечислять необязательно. Вместо этого можно сделать доступным всё содержимое пакета:

```
from math import *
```

Теперь можно использовать *все* функции пакета непосредственно

Демонстрация

<https://www.youtube.com/watch?v=mOQBZq9WCCY>

Первая программа

Установка PyCharm и создание первой программы:

<https://www.youtube.com/watch?v=LN5B0vkRhwwv&list=ru>

Outline

Введение в программирование

Блок-схемы

Язык программирования Python

IDE - интегрированные среды разработки

Интерактивный режим интерпретатора

Модули

Введение в язык

Литералы

Переменные

Операции

Определение типа

Ввод и вывод

Критерии качества ПО

Рекомендации и литература

Язык программирования

Язык программирования определяется:

- ▶ **Алфавит** – совокупность символов, которыми можно записывать программу
- ▶ **Синтаксис** – набор правил, описывающий комбинации символов алфавита
- ▶ **Семантика** – смысловое значение операторов, основных конструкций языка и т. п.

Алфавит языка Python

- ▶ Буквы национальных алфавитов: А-з, А-я, ...
- ▶ Символ подчёркивания: _
- ▶ Цифры: 0-9
- ▶ Специальные символы: + - * / > < = ; ' , . :
[] () @
- ▶ Комбинации символов – служебные слова, keywords
(считаются одним символом):
False None True and as assert async await break
class continue def del elif else except finally
for from global if import in is lambda nonlocal
not or pass raise return try while with yield

Лексемы

Лексема (token) – последовательность допустимых символов языка программирования, имеющая смысл для транслятора.

Виды лексем

- ▶ Идентификаторы (имена, identifiers)
Например имена переменных:
`x`, `x2`, `my_sum`, `iter_count`
- ▶ Служебные (зарезервированные слова, keywords)
- ▶ Литералы (неименованные константы или значения, literals)
Например:
`42`, `0`, `12.0`, `.32`, `"qwerty"`
- ▶ Знаки операций (punctuators)

Лексемы

Неправильные лексемы:

12abc

3.14.16

"qwerty

'qwerty

abc'def

(
}

@#\$\$%^&*\$

Комментарий

Комментарий — пояснение к исходному тексту программы, не влияющее на его семантику и находящееся непосредственно внутри комментируемого кода.

Комментарий игнорируется транслятором, но может быть полезен для программиста объясняя особенности алгоритма или структуры данных.

```
x = 1799  # год рождения Александра Сергеевича Пушкина
```

```
# считаю значение логистической функции
```

```
z = 1 / (1 + exp( - x) )
```

```
# todo: придумать формулу для подсчёта числа заболевших COVID
```

```
n = 0
```

Идентификаторы (имена, identifiers)

Идентификатор — это имя программного объекта: переменной, константы, массива, функции, класса и т. п.

Ограничения идентификаторов

- ▶ первый символ не должен быть числовым символом;
- ▶ первый символ может быть любой алфавитный символ (в том числе любой UTF-8 символ национальных алфавитов);
- ▶ далее в имени можно использовать как алфавитные, так и числовые символы, за исключением пробельных символов;
- ▶ в качестве имени нельзя использовать служебные слова;
- ▶ в python регистро-зависимые имена, т. е.: BookId, bookID, Bookid, bookid, bookId и т. д., — разные имена.

Идентификаторы. Примеры

Правильно

x

A

variable

my_var

my_var2

for_test

не рекомендуется

переменная

chislo

qwekjhsf

Неправильно

2x

1384

lambda

my var

my.var

my, var

Служебные слова (keywords)

Служебное слово - слово, имеющее специальное значение.

```
False    class    finally is    return
None     continue  for lambda try
True     def from    nonlocal  while
and del  global  not with
as elif  if or  yield
assert  else    import pass
break   except in  raise
```

Печать всех служебных слов языка Python:

```
import keyword
print(keyword.kwlist)
```

Outline

Введение в программирование

Блок-схемы

Язык программирования Python

IDE - интегрированные среды разработки

Интерактивный режим интерпретатора

Модули

Введение в язык

Литералы

Переменные

Операции

Определение типа

Ввод и вывод

Критерии качества ПО

Рекомендации и литература

Литералы

Литерал (Literal) — запись в исходном коде компьютерной программы, представляющая собой фиксированное значение.

Литералы классифицируются по типам значений.

- ▶ Целочисленные литералы

123

0

```
99999999999999999999 # точность не ограничена
```

```
0o177 # вшсьмиричное число (Oct)
```

0x9ff # Шестнадцатеричное число (heX)

0b101010 # двоичное число (Bin)

Литералы

► Литералы вещественного типа

1.23

1.

.123

3.14e-7 # 3.14 * 10 ^(-7)

4E210 # 4 10 ^210

4.0e+210 # 4.0 10210

Запись числа вида $3.14e-7$ называется *экспоненциальной записью* где e означает "умножить на 10 в степени xx".

$$3.14e-7 = 3.14 \cdot 10^{-7}$$

Литералы

- ▶ Литералы комплексного типа

```
3 + 4j          # 3 + 4i
3.0 + 4.0j      # то же самое
```

```
# можно писать большую J
3J              # 0 + 3i.
2 + 0j          #
```

Для обозначения мнимой части комплексного числа используют постфикс **j**.

Литералы

- ▶ Литералы логического типа

True

False

Регистр имеет значение.

Литералы

► Строковые литералы

'qwerty' # можно использовать одинарные кавычки
"abcdef" # а можно двойные

в двойных кавычках одинарные - это отдельный символ
"Can't"

и наоборот
'ФГБОУ ВО "ЗабГУ"'

"""текст в тройных кавычках может располагаться
на нескольких строках.
ещё одна строка
и ещё"""

Неправильно:

'кавычки должны соответствовать друг другу'

Литералы

- ▶ Пустое значение
`None`

Типы данных

Зачем нужны?

- ▶ Вся информация в компьютере хранится в виде набора 1 и 0.
- ▶ в виде 10110110 может быть закодирован символ, целое число или вещественное число (тогда часть цифр должны отвечать за мантиссу, а часть за порядок)
- ▶ Типы данных призваны различать способ кодирования информации,
- ▶ т.е. все данные (набор нулей и единиц) должны быть помечены типом данных, чтобы компьютер мог правильно их интерпретировать

Тип данных (тип):

- ▶ множество допустимых значений, которые могут принимать данные, принадлежащие к этому типу;
- ▶ набор операций, которые можно осуществлять над данными, принадлежащими к этому типу.

Типы

Простые типы

- ▶ NoneType
- ▶ bool
- ▶ int
- ▶ float

Составные типы

- ▶ str - строка
- ▶ tuple - кортеж
- ▶ list - список
- ▶ dict - словарь (отображение, ассоциативный массив)

Outline

Введение в программирование

Блок-схемы

Язык программирования Python

IDE - интегрированные среды разработки

Интерактивный режим интерпретатора

Модули

Введение в язык

Литералы

Переменные

Операции

Определение типа

Ввод и вывод

Критерии качества ПО

Рекомендации и литература

Переменные

Переменная (в Python) – имя, с которым может быть связано значение.

Объявление

Объявление (declaration) – указание идентификатора (и его свойств). Объявление используется, чтобы уведомить транслятор о существовании элемента.

Определение (definition) – одновременное указание идентификатора и его значения. Объявление используется, чтобы уведомить транслятор о существовании элемента и одновременно задать значение.

Переменные

В Python нельзя объявить переменную не задав ей значения потому, что без значения нельзя задать тип переменной

Т.е. идентификатор любой переменной должен быть определён.

Если переменную объявить нужно, но пока не ясно какое значение ей задать можно использовать пустое значение None:

```
a = None
```

Переменные

Переменные также как и литералы классифицируются по типам записанных в них значений.

Тип переменной определяется типом записанного в неё значения.

```
a = 42      # переменная целого типа
x = 3.14    # ... вещественного типа
s = "Hello, Python!" # ... строкового типа
b = True    # ... логического типа
```

Регистр имеет значение: A и a - разные переменные.

Символ = это оператор присваивания, он связывает имя переменной и значение ("записывает значение в переменную")

Переменные

Определим типы переменных (см. предыдущий слайд) с помощью функции `type`

```
type(a)  # int
type(x)  # float
type(s)  # str
type(b)  # bool
```

Функция `type()` принимает в качестве параметра переменную и возвращает её тип

Outline

Введение в программирование

Блок-схемы

Язык программирования Python

IDE - интегрированные среды разработки

Интерактивный режим интерпретатора

Модули

Введение в язык

Литералы

Переменные

Операции

Определение типа

Ввод и вывод

Критерии качества ПО

Рекомендации и литература

Операция, оператор и операнд

Оператор (statement) – элемент языка программирования задающий некоторое *действие*.

Примеры операторов:

+ – оператор сложения

> – оператор сравнения "больше"

Операнд – объект над которым производится операция.

Пример.

Операция сложения:

1 + 8

1,8 - операнды

Операции с вещественным типом

- ▶ Арифметические
+, -, *, // (целочисленное деление), % (остаток от деления), **
- ▶ Сравнение
== (равно), != (не равно), <, >, <=, >=
- ▶ Принадлежность диапазону

$x < y < z$

то же самое что и

$x < y$ **and** $y < z$

Операции

- ▶ Как и в математической записи в программе каждая операция имеет свой приоритет

$1 + 2 * 4$

- ▶ Приоритет можно менять с помощью скобок

$(1 + 2) * 4$

- ▶ Но в программе могут быть не только математические операции

Например оператор вывода на экран

```
print( (1 + 2) * 4 )
```

Оператор присваивания

```
x = 42 + 2
```

и другие операторы

```
x = float(input())
```

Как правило операторы выполняются слева направо и от самого вложенного

Операции

Как правило операторы выполняются слева направо и от самого вложенного

```
print( (1 + 2) * 4 )
```

1. $1 + 2$
2. $3 * 4$
3. `print(12)`

```
x = 42 + 2
```

Оператор присваивания формально описывается так
<переменная> = <выражение> . Если в выражении есть
какая-то операция то она будет вложена в оператор
присваивания и поэтому выполнится первой

1. $42 + 2$
2. $3 * 4$
3. `print(12)`

Операции с целым типом

- ▶ те же самые, что и с вещественным типом
- ▶ побитовые операции
 - » (побитовый сдвиг вправо)
 - « (побитовый сдвиг влево)

Outline

Введение в программирование

Блок-схемы

Язык программирования Python

IDE - интегрированные среды разработки

Интерактивный режим интерпретатора

Модули

Введение в язык

Литералы

Переменные

Операции

Определение типа

Ввод и вывод

Критерии качества ПО

Рекомендации и литература

Типы и преобразование типов

Какие типы будут у этих переменных?

```
x = 300 + 2.0  
y = 12 + 8 // 2  
z = sin( pi ) + 1  
a = 12.0 // 4  
b = 12.0 % 5
```

Преобразование типов

Какие типы будут у этих переменных?

```
x = 300 + 2.0      # float
y = 12 + 8 // 2     # int
z = sin( pi ) + 1   # float
a = 12.0 // 4       # float
b = 12.0 % 5        # float
```


Преобразование типов

Тип определяется по типу литерала.

Тип всегда приводится к наиболее общему.

Тип `float` является более общим чем `int`.

Тип `complex` является более общим для `float`.

Например при сложение вещественного и целого числа результат будет вещественного типа

Явное преобразование типов

`int(выражение)`

преобразует выражение в целое число

```
int("123") # строка -> целое, 123
int(5.125) # вещественное число -> целое.
# дробная часть отбрасывается
```

```
float("123") # строка -> вещесвत्व. число, 123.0
float(123)   # 123 -> 123.0
```

```
str(123)    # 123 -> "123"
str(5.125)  # 5.125 -> "5.125"
```

Outline

Введение в программирование

Блок-схемы

Язык программирования Python

IDE - интегрированные среды разработки

Интерактивный режим интерпретатора

Модули

Введение в язык

Литералы

Переменные

Операции

Определение типа

Ввод и вывод

Критерии качества ПО

Рекомендации и литература

Вывод данных

```
print( ... )  
print ( "Hello, Python!" )  
a = 10  
b = 3.14  
print("a = ", a, "; b = ", b)  
# будет напечатано:  
# a = 10; b = 3.14
```

Вывести на экран и не переходить на следующую строку:

```
print ( "Hello, Python!", end="" )
```

[pythoner.name/formatted-output](#) - форматирование вывода

Ввод данных с клавиатуры

Функция **input** ожидает ввод с клавиатуры и возвращает значение введенное пользователем.

```
s = input()
```

```
# в переменную s будут записаны введенные данные
```

Это значение имеет строковый тип (str), вне зависимости от того, что ввёл пользователь.

По окончании ввода пользователь должен нажать Enter.

Ввод данных с клавиатуры

Ввод пользователя необходимо преобразовывать к требуемому типу данных.

Например, чтобы преобразовать строку в целое и вещественное число нужно использовать явное преобразование типа:

```
d = int( input() )
```

```
f = float( input() ) # десятичный разделитель - точ
```

В этих примерах выполнение функции начинается с самого нижнего уровня вложенности: сначала `input`, затем результат выполнения функции `input` будет помещён в функцию `int`, а она уже преобразует его в значение целого типа.

Ввод данных с клавиатуры

Введённые данные при преобразовании должны быть корректными.

Например, невозможно преобразовать строки "data" или "три" в число с помощью функций `int` или `float`.

Если не удаётся преобразовать строку в число, то программа завершится аварийно с одной из следующих ошибок:

ValueError: invalid literal **for** `int()` **with** base 10

ValueError: could **not** convert string to `float`

Ввод данных с клавиатуры

Нужно однозначно сообщать пользователю смысл и формат входных данных, чтобы в большинстве случаев избежать ошибок при преобразовании значений.

При этом пояснения должны быть лаконичны.

```
print("Введите ваш вес в кг")
m = float( input() )
print("Введите ваш рост в метрах")
h = float( input() )

print("Индекс массы тела равен ")
print( m / h**2 )
```


Outline

Введение в программирование

Блок-схемы

Язык программирования Python

IDE - интегрированные среды разработки

Интерактивный режим интерпретатора

Модули

Введение в язык

Литералы

Переменные

Операции

Определение типа

Ввод и вывод

Критерии качества ПО

Рекомендации и литература

Критерии качества ПО

- ▶ **Надёжность.**

Программа должна корректно работать с любыми допустимыми входными данными, обрабатывать исключительные ситуации возникающие во время работы

Критерии качества ПО

► **Эффективность.**

Программа не должна совершать лишних действий (вычислений) и не должна хранить лишних данных.

Программы становятся медленнее куда шустрее, чем компьютеры становятся быстрее

– Никлаус Вирт

Критерии качества ПО

► Сопровождаемость.

Программа должна быть пригодна для дальнейшей модификации

Для этого должны быть соблюдены условия:

- хорошая читаемость кода (см. стандарт оформления кода)
- понятность алгоритма (комментарии, документация, названия переменных, моделей и функций)
- готовность к повторному использованию кода
созданные функции, модули и классы должны быть пригодны для использования при дальнейшей разработке или в других проектах
- модифицируемость кода
внесение изменений в логику работы программы или её дополнение не должны "ломать" всю программу

Программа должна совершенствоваться после её выпуска.
Поэтому её должно быть легко понять и легко вносить в неё изменения, в том числе сторонним программистам

Критерии качества ПО

- ▶ **Удобство использования**

Программа должна быть понятной, простой, должна решать задачи пользователя, а не возлагать на него новые

Outline

Введение в программирование

Блок-схемы

Язык программирования Python

IDE - интегрированные среды разработки

Интерактивный режим интерпретатора

Модули

Введение в язык

Литералы

Переменные

Операции

Определение типа

Ввод и вывод

Критерии качества ПО

Рекомендации и литература

Ссылки и литература. IDE

- ▶ youtube.com/watch?v=ln5B0vkRhww - Python и PyCharm(IDE) создание нового проекта
- ▶ youtube.com/watch?v=DpscmxH2LQU - Особенности и возможности PyCharm

Рекомендации по освоению курса

Больше практики!

Освоить программирование изучив только теорию невозможно.

Существующие курсы по программированию (университетские и онлайн-курсы) в часто позволяют освоить *только* основы программирования даже при выполнении всех предусмотренных заданий

Понимание основных концепций программирования, способов решения задач с помощью языка программирования и особенностей самих языков программирования приходит только с опытом

Рекомендации по освоению курса

Конспектирование и комментарии

Программирование - обширная область и сложная область. Каждый год появляется множество новые библиотеки и фреймворков для языков программирования, изменяются сами языки и появляются новые.

Самостоятельно написанный код месяц назад воспринимается как незнакомый. Поэтому комментарии в коде необходимы.

Блог или онлайн-заметки могут стать хорошим дополнением к конспекту, а так же способом поделится опытом или обсудить его

Рекомендации по освоению курса

Чужой опыт и лучшие практики (best practice)

Как лучше написать функцию? Какие функции вынести в модуль? Какой алгоритм лучше применить? Как лучше хранить данные? Какие структуры данных лучше использовать?

В программировании одну и ту же задачу можно решить несколькими способами. Некоторые из них плохие, некоторые хорошие.

best practice - общепризнанные методики решения задач.

Рекомендации по освоению курса

Резервное копирование

Лучший способ потерять код - хранить его только на флешке.
Второй по популярности способ - хранить только на своём компьютере.

- ▶ Регулярно делайте резервные копии.
- ▶ Используйте облачные хранилища (Google Drive, OneDrive, облака Яндекса и Mail.ru). Это способ не потерять результаты работы и иметь доступ к ним из любого места.
- ▶ Используйте систему контроля версии (например git) вместе с удалённым репозитарием (github, bitbucket, gitlab и др.)
- ▶ Сохраняйте рабочую версию программы перед внесением в неё значительных изменений

Рекомендации по освоению курса

Используйте лучшее. Изучайте новое.

Программирование - это сложно. Не позволяйте плохим программам сделать его ещё сложнее.

IDLE, блокнот Windows и cmd.exe почти наверняка испортят удовольствие от программирования.

Используйте текстовые редакторы с подсветкой синтаксиса (SublimeText, Atom) функциональные и удобные среды программирования. Используйте git. Делитесь кодом на pastebin и github. Сравнивайте используемые в отрасли программы на stackshare

Изучайте новые технологии. Попробуйте Linux (например Ubuntu).

Ссылки и литература

ЭБС

- ▶ biblio-online.ru - ЭБС Юрайт
- ▶ studentlibrary.ru - ЭБС "КОНСУЛЬТАНТ СТУДЕНТА"
- ▶ Федоров, Д. Ю. Программирование на языке высокого уровня python : учебное пособие для прикладного бакалавриата Содержит краткое описание языка.
- ▶ ru.wikibooks.org/wiki/Python - Викиучебник
- ▶ Лутц М. Изучаем Python. 2010. - 1280 с. Содержит подробное описание языка.
- ▶ Официальная документация Python3
`help(имя)`

Дополнительная литература

- ▶ O'Connor T.J. Violent Python: A Cookbook for Hackers, Forensic Analysts, Penetration Testers and Security Engineers. 2012 — 288 p.
- ▶ Numerical methods in engineering with Python 3 / Jaan Kiusalaas.

Полезные ресурсы

- ▶ stackoverflow.com - система вопрос и ответов о программировании (и не только)
- ▶ pastebin.com - веб приложение для быстрой загрузки фрагментов текста или кода

Для вдохновения :)

- ▶ Numb3rs
- ▶ Mr. Robot
- ▶ habrahabr: Игры, в которых нужно писать код

Ссылки и литература

Ссылка на слайды

github.com/VetrovSV/Programming